# AN5439
# アプリケーション・ノート

## ST25DV-I2C シリーズ・ダイナミック NFC タグによる拡張 NDEF

## はじめに

このアプリケーション・ノートの目的は、ST25DV-I2C シリーズのダイナミック NFC タグ（以下、ST25DV-I2C と呼ぶ）のパフォーマンスをマイクロコントローラによって向上させる方法、特にエンドユーザに拡張 NDEF 体験を提供する方法を説明することです。

ST25DV-I2C デバイス（すなわち、ST25DV04/16/64K および ST25DV04/16/64KC）は、4/16/64Kbit 容量の EEPROM を内蔵しています。

拡張 NDEF は NFC フォーラム・タグに格納された NDEF（NFC データ交換フォーマット）メッセージの略で、その内容はタグによって動的に変更され、追加情報が提供されます。

NDEF メッセージは ST25DV-I2C に保存され、NFC フォーラム互換デバイスによって読み取られます。これらのメッセージには、URI、テキスト、画像、Bluetooth® 接続などの情報が含まれています。

NFC フォーラム対応スマートフォンによって、NDEF メッセージが読み取られると、外部アプリケーションなしに、URL を含む NDEF メッセージに対してウェブブラウザを開いたり、Bluetooth ペアリングを自動的に実行したりするなど、特定のアクションが自動的にトリガされます。

NFC フォーラム・タグ内の NDEF メッセージの内容は通常静的です。NFC フォーラム・リーダやスマートフォンによって変更されますが、外部イベントによって自動的に変更されることはありません。外部または内部のイベントに応じて NDEF メッセージの内容が動的に更新されるのは興味深いことです。たとえば、URL に動的な情報を埋め込んで、ユーザをパーソナライズされたウェブページに自動的にリダイレクトしたり、SMS や E メールにセンサ・データを埋め込んで、タグをタップしたときにセンサのデータを自動的に送信したりすることができます。NDEF メッセージの読取りを Android™ と iOS がネイティブサポートしているため、これらのアクションは追加のアプリケーションなしに実行されます。

# 目次

# 表の一覧

# 図の一覧

# 1 概要

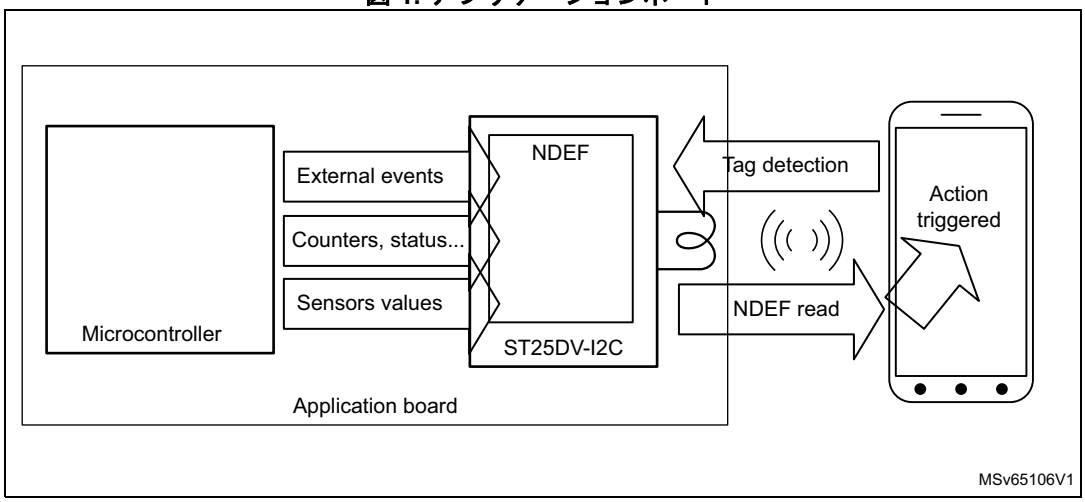
EEPROM の内容は、RF インタフェースを通じてスマートフォンまたは NFC フォーラム・リーダから、また有線 I2C インタフェースを通じてマイクロコントローラからアクセスされます。

NDEF メッセージは、ST25DV-I2C のユーザ・メモリに格納されます。

スマートフォンが ST25DV-I2C に接続されたアンテナに十分に近づくと、タグの存在が検出され、ST25DV-I2C メモリに保存されている NDEF メッセージの読取りが開始されます（図 1 を参照）。

NDEF メッセージの内容は通常は静的ですが、マイクロコントローラによってバックグラウンドで、またはスマートフォンが近づいたときに動的に、NDEF メッセージの内容が更新され、NDEF メッセージには常に動的なデータが含まれるようになります。

**図 1. アプリケーションボード**




NDEF メッセージには、次のようないくつかのデータが動的に付加されます。

- 識別情報（UID、署名など）
- タップ・カウンタ
- 改ざんステータス
- MAC：Message Authentication Code（メッセージ認証コード）
- センサの値（温度、湿度、圧力、距離、電圧など）
- GPS 位置
- 日時
- バッテリ残量
- 診断ログ

Android および iOS ベースのスマートフォンは NDEF メッセージをネイティブにサポートしているため、スマートフォンが NDEF メッセージを読んでいる間に、ブラウザを開く、E メールやテキスト・メッセージを送信する、アプリケーションを開くなどのアクションが自動的にトリガされます。

# 2 拡張 NDEF の内容の更新

NDEF メッセージの内容はマイクロコントローラによって以下の 3 つのフェーズで更新されます。

フェーズ 1 および 2 :

1. スマートフォンの存在が検出され、スマートフォンが NDEF メッセージの読取りを開始する前
2. スマートフォンによる NDEF の読取り中

フェーズ 3 :

3. それ以外の、スマートフォンが NDEF メッセージを読み取ろうとしていないとき

最初の 2 つのフェーズは NDEF メッセージの「オンデマンド更新」です。スマートフォンが NDEF メッセージを読もうとするたびに、コンテンツは「ジャスト・イン・タイム」で更新されます。

3 つ目は、スマートフォンの存在とは非同期です。これは、NDEF メッセージの定期的な更新や、他の外部イベント（センサからの割込みなど）によって引き起こされる更新に対応します。

非同期方式では特定の技術を伴わないので、以降のセクションでは「オンデマンド更新」方式に焦点を当てます。

## 2.1 スマートフォン検出時の NDEF 内容の自動更新

スマートフォンの検出時に NDEF メッセージの内容を更新することには、いくつかの利点があります : データは常に新しく、アプリケーションはバッテリ・レスで実行できます（エナジー・ハーベスティングによって電力を供給できるほど消費電力が低いと仮定して）。

しかし、この方法は重要なタイミング制約が必要となるため、最も難しい方法です。更新するデータ量によっては、実行できないこともあります。

タイミング制約を理解するには、まず ST25DV-I2C で I2C と RF インタフェース間のアービトレーション（調停）がどのように行われるか、そしてスマートフォンがどのようにタグの存在を検出して NDEF メッセージを読み取るのかを理解する必要があります。

### 2.1.1 RF と $I^2C$ インタフェース間のアービトレーション

スマートフォン検出時に NDEF メッセージを更新する場合、マイクロコントローラ（$I^2C$ インタフェース経由）とスマートフォン（RF インタフェース経由）の両方が ST25DV-I2C にアクセスしようとします。マイクロコントローラは EEPROM に書き込もうとし、スマートフォンは EEPROM を読み出そうとします。

その性質上、これら 2 つのホスト・コントローラは同期していないため、ST25DV-I2C に同時にアクセスしようとします。このような状況を管理するために、ST25DV-I2C は RF 側と $I^2C$ 側からの同時通信を処理するアービトレーション回路を内蔵しています。

アービトレーションは「早い者勝ち」の原則に基づいています。これは、I²C チャネルと RF チャネルがビジー状態であるかどうかによって決まります。

- ST25DV-I2C は、I²C コマンドをデコードして実行しているとき、および有効な I2C 書込みコマンドに続いて EEPROM をプログラミングしているときに、I²C ビジー状態になります。
- ST25DV-I2C は、RF コマンドをデコードして実行しているときに RF ビジー状態になります。

両方のインタフェースがアクティブな場合、ST25DV-I2C は最初に受信したコマンドをデコードして実行します（表 1 を参照。通信アービトレーションの起こり得るケースを説明しています）。

表 1. 通信アービトレーションの起こり得るケース

| 初期状態のイベント・アクション | イベント | アクション |
|---|---|---|
| ST25DV-I2C は I²C ビジー状態：<br>I²C コマンドをデコード中または実行中 | I²C コマンド処理中に<br>RF コマンドが送信される | RF コマンドはデコードされない(1) |
| ST25DV-I2C は RF ビジー状態：<br>RF コマンドをデコード中または実行中 | RF コマンド処理中に<br>I²C コマンドが送信される | I²C コマンドはデコードされない(2) |

1. RF コマンドのインベントリは停止したままで、UID（固有の識別子）が一致するアドレス指定された RF コマンドは応答を受信しません。その他の RF コマンドはエラー・コード 0Fh を受信します。
2. I²C マスタは、最初のデータ・バイト（つまり、デバイス選択）の 9 ビット目でスレーブの「No Ack」状態を受信します。

スマートフォン検出時に NDEF メッセージを更新する際の主な課題は、表 1 の 1 行目に示すように $I^2C$ がビジー状態のときに、スマートフォンに RF リクエストを送信させないようにすることです。最終的に、スマートフォンが RF リクエストでエラーを受信した場合（または応答がない場合）、通常は NDEF の読取り手順を中断することで、これを避ける必要があります。

### 2.1.2 スマートフォンでの NDEF 検出と読取りの手順

スマートフォン（およびすべての NFC フォーラム準拠のリーダ）では、NDEF メッセージの読取りはタグ検出と NDEF 読取りの 2 つの主要なフェーズに分かれています。

#### タグ検出フェーズ

タグ検出フェーズでは、スマートフォンはまず 13.56MHz の短い無変調パルスを発信し、フィールド（電磁場）内の 13.56MHz 対応オブジェクトの存在を検出します。

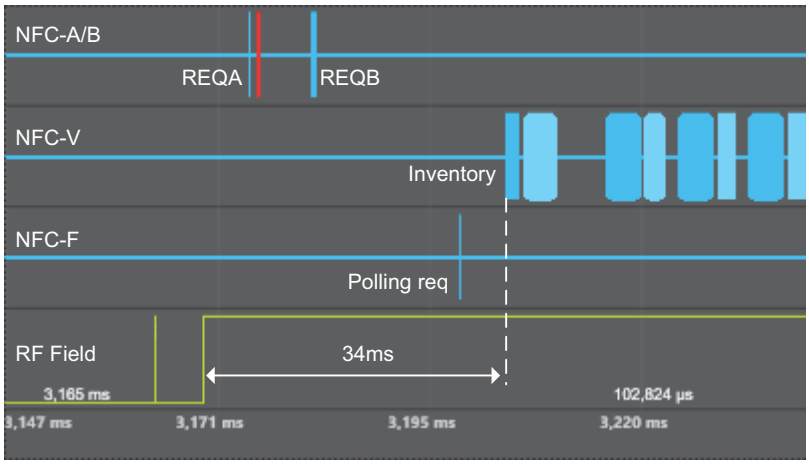
存在が検出されると、スマートフォンは連続的な 13.56MHz の RF フィールドの放射を開始し、サポートされているすべての NFC フォーラム・テクノロジーに対してアンチ・コリジョン手順を開始します（このステップは NFC フォーラムの「アクティビティ」仕様に記述されており、テクノロジー検出アクティビティと呼ばれます）。

ほとんどのスマートフォン（Android と iOS）は NFC フォーラムの NFC-A、NFC-B、NFC-F、NFC-V テクノロジーをサポートしており、それらに対してポーリングします。スマートフォンの中には、ACM（NFC-A または NFC-F、あるいはその両方でのアクティブ通信モード）をサポートし、ポーリングするものもあります。

NFC-V 検出とアンチ・コリジョンは常に最後に実行されます。つまり、RF フィールドの立ち上がりと NFC-V アンチ・コリジョン（インベントリ・コマンド）の開始には若干の遅れがあります。この遅延は通常約 35ms ですが、スマートフォンによって異なる場合があります。4 つの NFC-A/B/F/V テクノロジー（NFC フォーラム仕様で定義されている）がサポートされている場合、遅延は 35ms より大きくなることがありますが、30ms より小さくなることはありません。

図 2、図 3、図 4 は、RF スパイ・アナライザで取得した、異なる OS を搭載した 3 台のスマートフォンのタグ検出フェーズを示しています。各 NFC テクノロジーと RF フィールドの状態に対してスマートフォンから送信されたコマンドが、RF スパイによって記録されています。

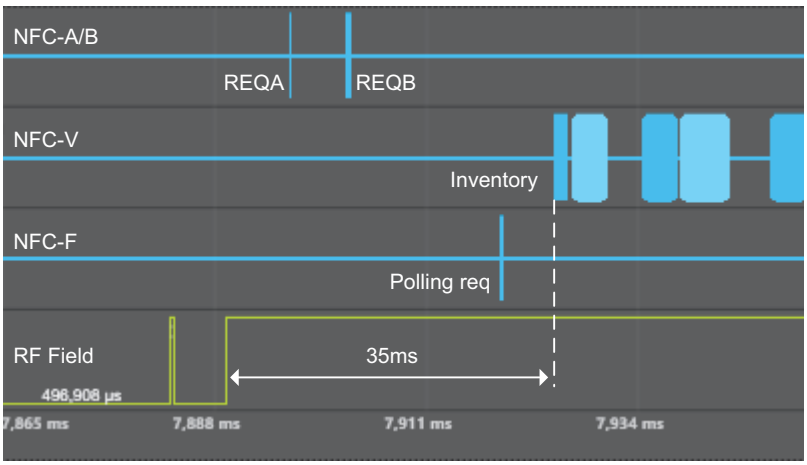
**図 2. スマートフォン 1 のタグ検出フェーズ**




スマートフォン 1 は以下のようにタグの存在を検出し、次に NFC テクノロジーを検出しようとします。

- スマートフォンは、まず 13.56MHz の非常に短いパルスを無変調で約 40µs 間送信します。
- 次に RF フィールドは 5.4ms 間オフになります。
- この一時停止の後、NFC テクノロジーの検出が開始されます。最初に NFC-A、次に NFC-B、NFC-F、最後に、RF フィールドの立ち上がりから 34ms 後に NFC-Vが検出されます。
- 送信される最初の NFC-V コマンドはインベントリ・コマンド（アンチ・コリジョン）です。

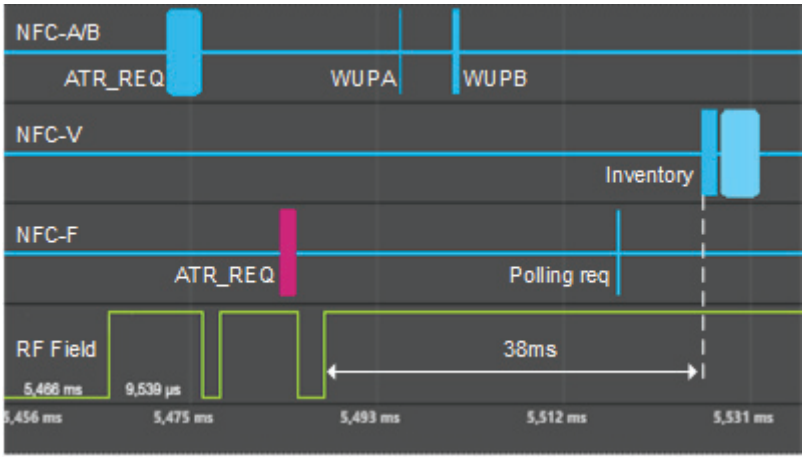
**図 3. スマートフォン 2 のタグ検出フェーズ**




スマートフォン 2 のタグ検出フェーズはスマートフォン 1 のそれに非常に似ています。次のようにタイミングが若干異なります。

1. 最初の短い 13.56MHz の無変調パルスの期間は約 320µs です。
2. NFC テクノロジーの検出前の一時停止は約 5.7ms です。
3. NFC-V のアンチ・コリジョンは RF フィールドの立ち上がりから 35ms 後に開始します。

図 4. スマートフォン 3 のタグ検出フェーズ




スマートフォン 3 のタグ検出フェーズは、NFC ピア・ツー・ピア・デバイス（ACM）を検出するために、変調された 13.56MHz の RF フィールドの 2 パルスで始まり、それぞれ一時停止が続きます。その後、手順はスマートフォン 1 やスマートフォン 2 で見られたものと同じです。タイミングは以下のとおりです。

1. 13.56MHz のパルス：9.5ms、その後 1.8ms の一時停止
2. 13.56MHz のパルス：7.8ms、その後 2.78ms の一時停止
3. RF フィールドの立ち上がりから 38ms 後に NFC-V のアンチ・コリジョンによる NFC タグ検出

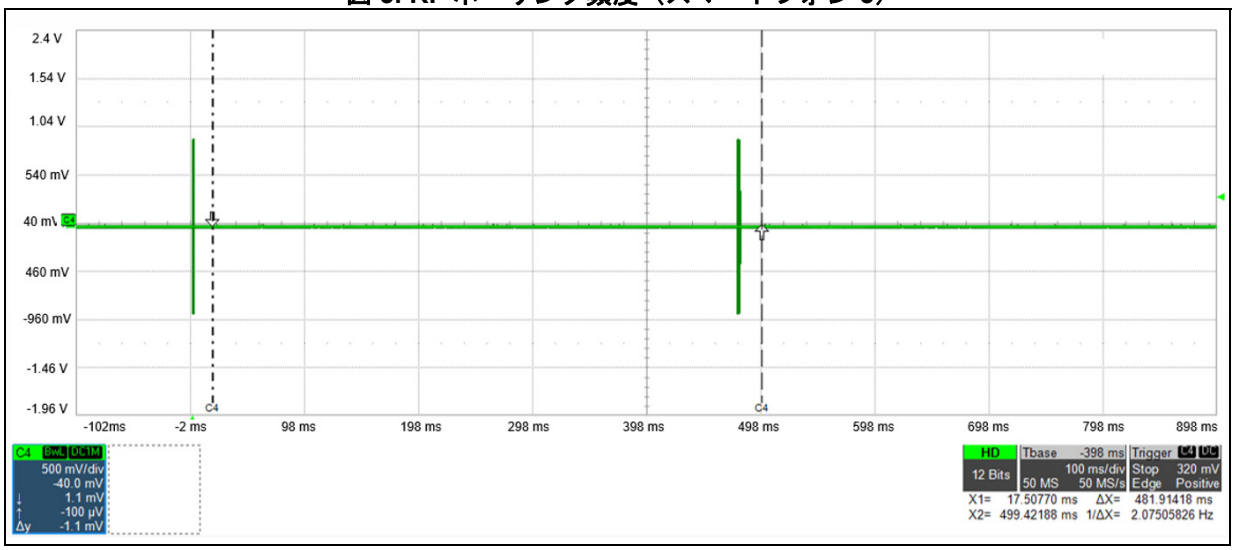
これらの 3 つのスマートフォンの例では、搭載している OS が何であれ、一般的に起こることが要約されています。

NFC-V のアンチ・コリジョンを開始する前の最小 30ms の遅延によって、マイクロコントローラは ST25DV-I2C メモリ内の NDEF メッセージの内容を更新することができます。ST25DV-I2C は、RF フィールドが立ち上がるとすぐに、エナジー・ハーベスティングによってマイクロコントローラをブートするための電力を供給することができ、その後、マイクロコントローラは最初の NFC-V コマンド（インベントリ）の前 30ms の間に、$I^2C$ を介して NDEF メッセージを更新します。この方法をセクション 2.1.4 に詳しく示します。

NFC-V アンチ・コリジョン・フェーズの後、タグが見つからなかった場合（インベントリ・コマンドに対する応答がなかった場合）、スマートフォンはバッテリを節約するために RF フィールドの放射を停止します。インベントリ・コマンドの期間は 2.2ms です。

その後、スマートフォンは図 5 に示すように、通常約 500ms のポーリング頻度で新しいポーリング・フェーズを開始します。

図 5. RF ポーリング頻度（スマートフォン 3）




NDEF メッセージの更新に必要な時間が 30ms を超える場合、マイクロコントローラは、最初のポーリング・フェーズでの検出時に NDEF メッセージの内容を更新してからポーリング・フェーズの使用を有効にし、その後のポーリング・フェーズ時でのみタグがスマートフォンに応答できるようにします。ST25DV-I2C の GPO_FIELD_CHANGE 割込みによって RF フィールドが検出されます。したがって、NDEF メッセージの更新に割り当てられる時間は、ユーザが許容する応答時間によってのみ制限されます。しかし、各ポーリング・フェーズの間に RF フィールドが存在しないため、マイクロコントローラの電源にエナジー・ハーベスティングを使用することはできず、バッテリまたは常時電源が必要となります。この方法をセクション 2.1.5 に詳しく示します。

### NDEF 読取りフェーズ

NFC-V アンチ・コリジョンが実行された後、スマートフォンは NDEF 読取り手順を開始します。この手順はタイプ 5 タグに関して NFC フォーラムによって指定されています。

インベントリ・コマンドの後、すぐに CC ファイルの読取りから手順が開始されます。すなわち、スマートフォンはリード・シングル・ブロック・コマンドを送信し、タグの最初の 4 バイト（または 8 バイト）を読み取ります。その後、スマートフォンは NDEF メッセージそのものを読み取るために、複数のリード・シングル・ブロック・コマンドまたはリード・マルチプル・ブロック・コマンドを送信します（CC ファイルの後に素早く開始します）。

Android スマートフォンは、タグ検出のインベントリ手順と NDEF 読取り手順の間に、システム情報の取得と複数ブロックのセキュリティ・ステータスの取得という ISO15693 コマンドを追加します。

NFC フォーラムの NFC-V テクノロジーでは、2 つの連続する RF コマンド間の最小タイミング（応答終了から新たな要求開始までの時間）は 309µs です。実際には、スマートフォンの 2 つの RF コマンド間の遅延はもっと大きく、典型的な値は 1.5～5ms の間です。この遅延は通常、ST25DV-I2C の EEPROM への I2C 書込みアクセスを可能にするには短すぎます。スマートフォンによる NDEF 読取り中に NDEF メッセージの内容を更新することは、有効な選択肢ではありません。実際、新しい RF コマンドを開始する前に I2C リクエストが終了していないと、RF エラー応答が発生し、スマートフォンによる NDEF 読取りプロセスが即座に停止する可能性があります。

図 6. スマートフォン 2 の NDEF 読取りフェーズ

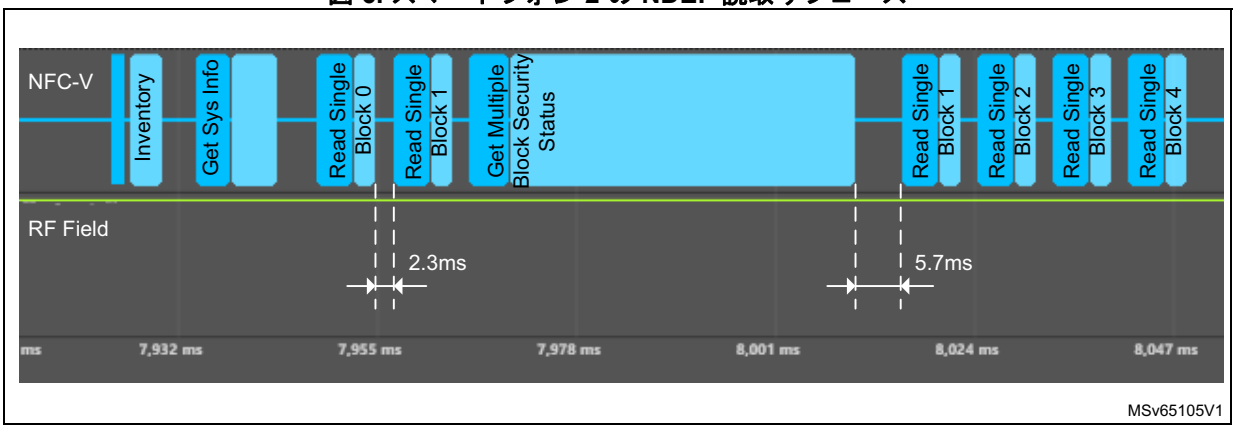
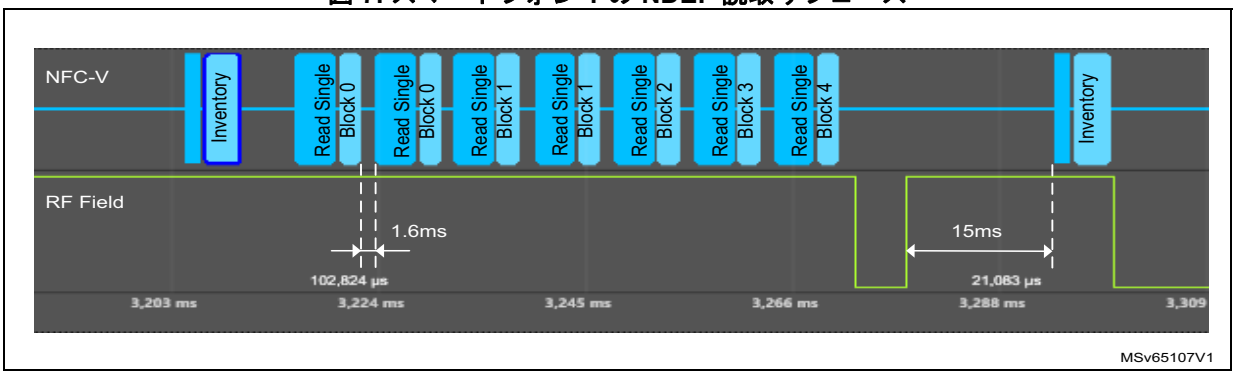



図 7. スマートフォン 1 の NDEF 読取りフェーズ




スマートフォン 1 は、NDEF 読取りの終了後に RF フィールドの放射を停止し、その後定期的に RF を再アクティブ化し、タグがまだ存在することを確認するためにインベントリ・コマンドを送信します。

### 2.1.3 NDEF の内容の自動更新のタイミング制約

制約は、RF コマンドの失敗を回避し、スマートフォンからの中断を回避することです。つまり、$I^2C$ コマンド実行中（EEPROM プログラミング時間を含む）に RF コマンドを受信しないということです。

NDEF メッセージを更新するためには、マイクロコントローラは ST25DV-I2C の EEPROM にデータを書き込む必要があります。ST25DV-I2C のプログラミング時間は、4 バイトのブロックあたり 5ms です（すべてのバイトが同じ行（ページとも呼ばれる）に属していると仮定）。

NDEF 読取りフェーズで説明されているように、NDEF 読取りフェーズ中の 2 つの連続する RF コマンド間の遅延は 5ms 未満であるため、RF コマンドとのアービトレーション競合を起こさずに、このフェーズ中に ST25DV-I2C の EEPROM に $I^2C$ からデータを書き込むことはできません。

したがって、RF と $I^2C$ 間の衝突を避けるには、RF フィールドの立ち上がりからアンチ・コリジョンの開始（最初のインベントリ・コマンド）までの間に、マイクロコントローラによって NDEF の更新を完了させる必要があります。この時間は通常ほとんどのスマートフォンで少なくとも 30ms あります。

NDEF の更新タイミングは以下の 3 つのフェーズに分かれます。

1. マイクロコントローラがウェイクアップまたはブートする
2. マイクロコントローラが、ST25DV-I2C のデータ（通常はカウンタ値）を、または外部デバイス（通常はセンサ）のデータを読み込む可能性がある
3. マイクロコントローラが ST25DV-I2C の EEPROM に書き込む（NDEF の内容の更新）

マイクロコントローラのウェイクアップとブートのフェーズは、マイクロコントローラによって異なるため、このアプリケーション・ノートでは詳しく説明しません。

データ読取りフェーズの速度は、基本的に I2C バスの周波数と、外部デバイスへのアクセスが必要か（つまり、変換時間の長いセンサがあるか）どうかに依存します。

ST25DV-I2C メモリの更新速度は、主に EEPROM のプログラミング時間に依存します。$I^2C$ の書込みの時間を計算するには、$I^2C$ 書込みコマンドを 2 つの部分、すなわち、バス上で送信される $I^2C$ のバイト数と、STOP 条件の直後に始まる EEPROM プログラミングフェーズとに分割することができます。

- I2C 書込みコマンドの時間 = (3 + 書込みバイト数) / (9 * I2C クロック周期)
- EEPROM プログラミングの時間 = $5 * 10^{-3}$ * (プログラミングするページ数)
- I2C 書込みの合計時間 = I2C 書込みコマンドの時間 + EEPROM プログラミングの時間

ST25DVxxK デバイスの EEPROM ページは 4 バイト長です。同じページにあるデータはアドレス・ビット b16～b2 を共有します。時間的には、1 バイトのプログラミングは 1 ページのプログラミングに相当し、5 バイトのプログラミングは 2 ページのプログラミングに相当します。

たとえば ST25DV04K の場合、$I^2C$ クロックが 400kHz であれば、アドレス 000Ch から始まる 16 バイトを 20ms で書き込むことができます。アドレス 000Bh から始まる場合、またはアドレス 000Ch から始まる 17 バイトを書き込む場合、動作にかかる時間は 25ms です。

ST25DVxxKC では、EEPROM ページは 16 バイト長です。同じページにあるデータはアドレス・ビット b16～b4 を共有します。時間的には、5 バイトのプログラミングは 1 ページのプログラミングに相当し、17 バイトのプログラミングは 2 ページのプログラミングに相当します。

たとえば ST25DV04KC では、I2C クロックが 1MHz であれば、アドレス 0010h から始まる 16 バイトを 5ms で書き込むことができます。アドレス 0011h から始まる場合、またはアドレス 0010h から始まる 17 バイトを書き込む場合、動作にかかる時間は 10ms です。したがって、NDEF メッセージで更新可能なデータ量は限られており、プログラム開始アドレスとその結果としてのページ数には特別な注意を払う必要があります。

更新されるデータの長さが 4 バイトより大きい場合は、ST25DVxxK よりも ST25DVxxKC デバイスの使用を優先すべきです。

### 2.1.4 バッテリ・レスの NDEF 内容自動更新の実装

前のセクションで説明したように、ST25DV-I2C ダイナミック・タグとマイクロコントローラを使用して、拡張 NDEF アプリケーションを作成することで、バッテリ・レスのアプリケーションを作成することができます。

アプリケーションは、スマートフォンによる NDEF メッセージの正しい読取りを可能にするために、スマートフォンが RF フィールドを提供し始めるとすぐに、そして 30ms 以内に NDEF メッセージを更新する必要があります。

原則として、この拡張 NDEF アプリケーションは次のように動作します。

- 標準の NFC タグとして動作する
- RF フィールドが立ち上がっている時にマイクロコントローラを起動する
- エナジー・ハーベスティングを利用して、マイクロコントローラと、場合によっては他のオンボード外部デバイスに電力を供給する

– ST25DV-I2C と、場合によっては他のオンボード外部デバイスのデータを読み取る
– ST25DV-I2C のメモリにデータを書き込み、NDEF メッセージの内容を更新する
– このすべてを、最初の NFC-V コマンドの前に（30ms 未満で）実行する

これを達成するための前提条件は次のとおりです。

1. 最初の NDEF メッセージは、ST25DV-I2C メモリに事前にプログラムされている。
2. ST25DV-I2C は、ブート時に EH が有効になるように設定されている（EH_MODE 設定レジスタ = 00h）。
3. 最小限のコンポーネントと低消費電力のコンポーネントが使用されている。

最大 30ms のタイミングを達成するためには、EH の電流供給開始時のコンデンサ充電、MCU ブート時間、データ読取り、データ書込みなど、すべてのフェーズを時間最適化する必要があります。ティアリングの問題を避けるために、特別な注意が必要です。

EEPROM へのデータ書込み中に、RF フィールドが失われる可能性があります（スマートフォンが取り外された場合など）。その場合のデータ破損を避けるためには、ティアリング防止機構を導入する必要があります。

簡単な解決策は、カウンタを書き込むことで NDEF の更新を終了し、このカウンタだけをティアリングから保護することです。NDEF データの有効性は、カウンタの値が前の値より高ければ（カウンタがインクリメントされていれば）保証されます。

カウンタをティアリングから保護するために、コンデンサに蓄えられたエネルギーが使われます。セクション 2.1.3 で説明されたように、EEPROM のプログラミングは $I^2C$ 書込みコマンドの停止条件の後にのみ開始されます。EEPROM のプログラミング中、マイクロコントローラは $I^2C$ クロックやその他の信号を提供する必要がないため、ST25DV-I2C にのみその $V_{cc}$ ピンを介して電力を供給する必要があります。最後の $I^2C$ 書込み（カウンタの更新）の停止条件の後までのみ ST25DV-I2C の電力供給を維持することで、消費電力は最小限に抑えられ、RF フィールドが除去された場合、ST25DV-I2C はコンデンサの放電によって電力供給されます。これにより、カウンタ更新のティアリング防止が達成されます。

定電流でのコンデンサ放電は、C = I * dt / dV として計算されます。I = 350µA（3.3V、125°Cにおける最大値）、dV = 3V - 1.8V（3V は標準的な EH 出力値、1.8V は ST25DV-I2C の最小動作電圧）、dt = 5ms（標準的なページ書込み時間）と仮定すると、ST25DV-I2C がページを書き込むために必要な最小の C 値は 1.46µF となります。カウンタが 1 ページ（ST25DVxxK では 4 バイト、ST25DVxxKC では 16 バイト）を超える場合、dt は 1 ページあたり 5ms のステップで増やす必要があります。

EH 電源が電力供給を開始する際に、1.8V に達するまでの充電時間が長すぎないように、コンデンサの値とコンデンサの充電時間のバランスをとる必要があります。NDEF の総更新タイミングを 30ms 以下に保とうとする場合、コンデンサの充電時間を考慮しなければなりません。

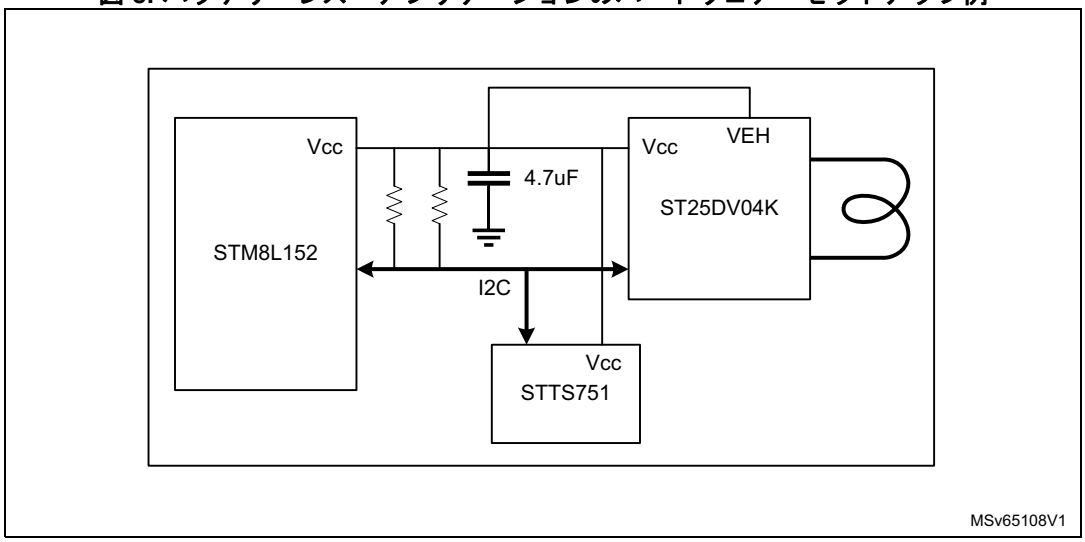
### ST25DV04K によるバッテリ・レスの NDEF 内容自動更新の例

この例では、拡張 NDEF タグから URL を含む NDEF メッセージが提供されます。URL の終わりは、センサで読み取られた温度値と、NDEF を読み取るたびにインクリメントされるタップ・カウンタによって自動的に完了します。

ハードウェアのセットアップ：

– NDEF メッセージがプリロードされ、ブート時に EH が有効にされる ST25DV04K ダイナミック・タグ
– STM8L152 超低電力マイクロコントローラ
– STTS751 低電力・高速変換時間・温度センサ
– 2 x 4.7µF コンデンサ（デバイスのデカップリング用に 10nF と 100nF のコンデンサを追加）

**図 8. バッテリ・レス・アプリケーションのハードウェア・セットアップ例**




NDEF メッセージは以下のデータを含む URL タイプ（example.com/temp=0000/tapcounter=0000）です。

- 温度値およびタップ・カウンタ値のアドレスは、プログラミング速度を向上させるために（各 5ms）、EEPROM ページに揃えられ、わずか 4 バイト長となっています。
- 温度とタップ・カウンタは NDEF メッセージの中に ASCII 形式で書き込まれています。
- 10 進数なら 10000、16 進数なら 65536、すべての英数字値なら $1.6 * 10^6$ の値が得られるので、4 文字のタップ・カウンターで十分です。

**表 2. タグのメモリ内容（ST25DV04K）**

| バイト・アドレス | バイト値 | ASCII | コメント |
|---|---|---|---|
| 0000 | E1 40 40 00 | á@@. | CC ファイル |
| 0004 | 03 2A D1 01 | .*Ñ. | NDEF TLV |
| 0008 | 26 55 01 65 | &U.e | URL 値 |
| 000C | 78 61 6D 70 | xamp | - |
| 0010 | 6C 65 2E 63 | le.c | - |
| 0014 | 6F 6D 2F 74 | om/t | - |
| 0018 | 65 6D 70 3D | emp= | - |
| 001C | 30 30 30 30 | 0000 | 温度値 |
| 0020 | 2F 74 61 70 | /tap | - |
| 0024 | 63 6F 75 6E | coun | - |
| 0028 | 74 65 72 3D | ter= | - |
| 002C | 30 30 30 30 | 0000 | タップ・カウンタ値 |
| 0030 | FE 00 00 00 | Þ… | TLV ターミネータ |

マイクロコントローラで（ST25DV04K を使用して）実行されるサンプルコード：

```
/* initialize I2C bus and clocks */
I2C_Init();
/* Read I2C sensor to get current temperature */
GetOneTemperature (&data_sensor);
```

```
/* update temperature value in NDEF message */
ConvertTempToAscii(data_sensor, data_char);
I2C_WriteOnePage(ST25DV_ADDRESS_USER, temperature_addr, data_char);
/* poll for end of EEPROM programming (~5ms) */
while( I2C_Poll(ST25DV_ADDRESS_USER) );
/* read current tap counter in tag's EEPROM */
I2C_ReadBuffer(ST25DV _ADDRESS_USER, tapcounter_addr, 4, data_char);
/* increment tap counter and update value in NDEF message */
IncrementTapCounterAscii(data_char);
I2C_WriteOnePage(ST25DV_ADDRESS_USER, tapcounter_addr, data_char);
/* immediately stop MCU to reduce power consumption */
halt();
```

ブート時間を短縮するため、初期化フェーズは最小限に抑えられています。$I^2C$ 書込み（タップ・カウンタ）が終了すると、マイクロコントローラは消費電力を最小化するために停止モードに設定されます（STTS751 はすでに最小電力モードになっています）。これにより、RF フィールドがオフになった場合、コンデンサに蓄積されたすべてのエネルギーが ST25DV-I2C で利用できるようになります（タップ・カウンタ更新のティアリング防止）。

図 9 と図 10 は、スマートフォン 1 とスマートフォン 4 での NDEF 更新動作を示すオシロスコープのスクリーンショットです。

**図 9. スマートフォン 1（ST25DV04K）によるバッテリ・レス NDEF 更新のキャプチャ図**

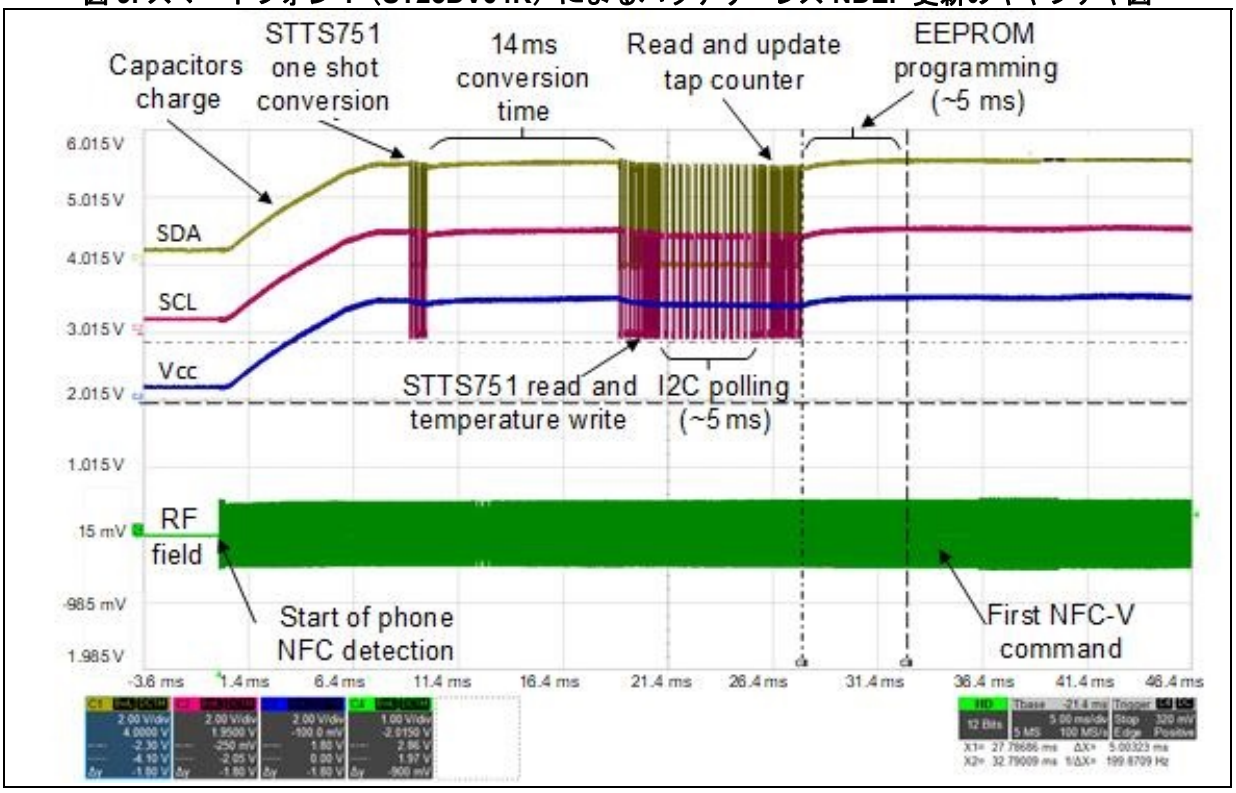
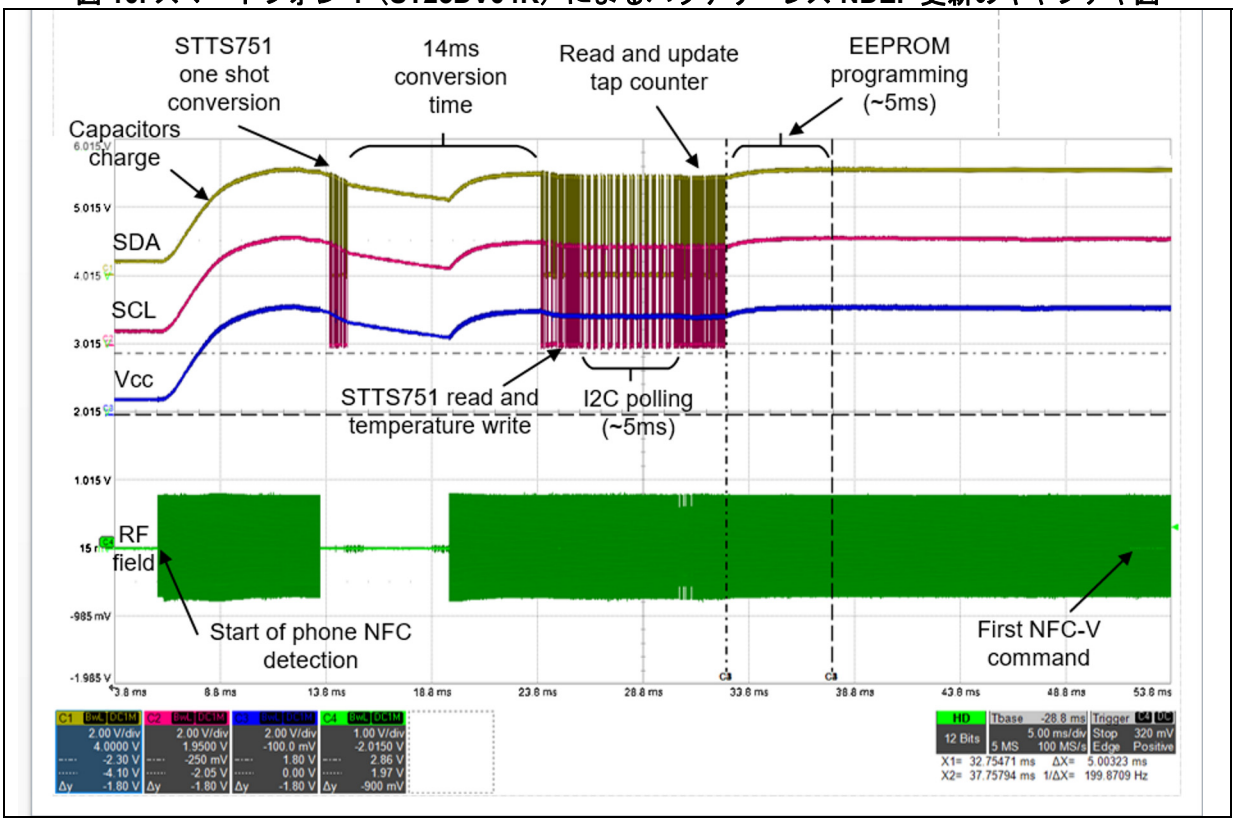


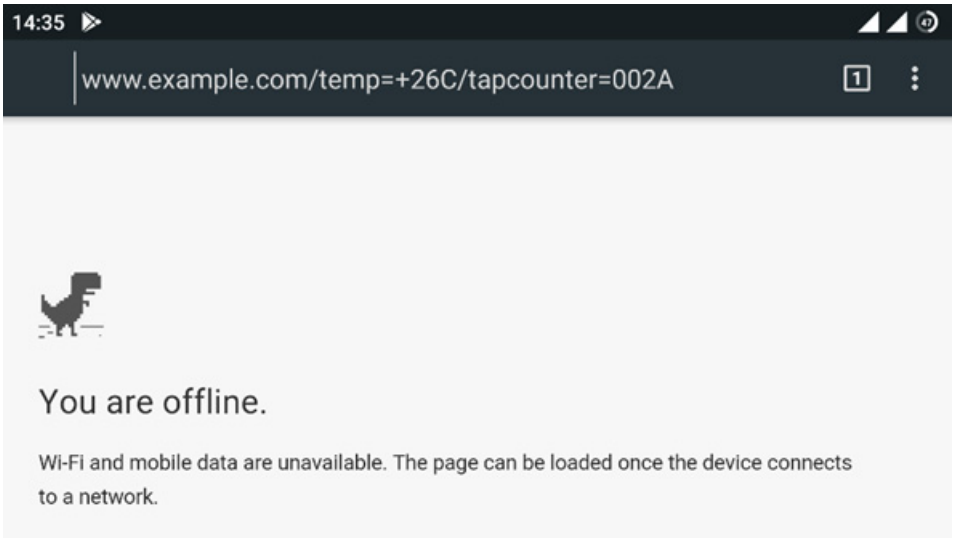
図 10. スマートフォン 4（ST25DV04K）によるバッテリ・レス NDEF 更新のキャプチャ図




スマートフォン 4 では、最初の ACM 検出の後、RF フィールドは約 6ms 間オフになりますが、2 x 4.7µF のコンデンサは $V_{cc}$ 電圧を 1.8V 以上に維持するのに十分です。

タグのアンテナにスマートフォンをかざすと、スマートフォンは自動的にウェブブラウザを開き、図 11 に示すように、温度（+26℃）とタップ回数（2Ah、42 回）に関する動的情報を含む URL が表示されます。

図 11. スマートフォン 4（ST25DV04K）でのバッテリ・レス NDEF 更新の結果例



### ST25DV04KC によるバッテリ・レスの NDEF 内容自動更新の例

ST25DV04KC は 5ms で $I^2C$ から 16 バイトを書き込めますが、ST25DV04K は同じ 5ms で 4 バイトしか書き込めないため（データがページに揃えられていると仮定）、ST25DV04K を ST25DV04KC に置き換えるメリットがあります。これにより、NDEF メッセージをより高速に更新したり、より多くのデータを更新したりすることが可能になります。

ハードウェアのセットアップは同じで、ST25DV04K が ST25DV04KC に置き換わるだけです（ピン配置互換）。

NDEF メッセージは、更新されるデータを ST25DV04KC のページに揃えるために若干変更されます（example.com/current_temp=0000/tc=0000）。ST25DV04K では 5ms の 4 バイトの書込みが 2 回必要であったのに比べ、この場合、5ms の 16 バイトの書込み 1 回で温度値とタップ・カウンタ値を更新することができます。

温度値とタップ・カウンタ値は同じページ（アドレス 0020h から始まる 16 バイト）にあるので、1 回の I2C 書込みコマンドで更新されます。

**表 3. タグのメモリ内容（ST25DV04KC）**

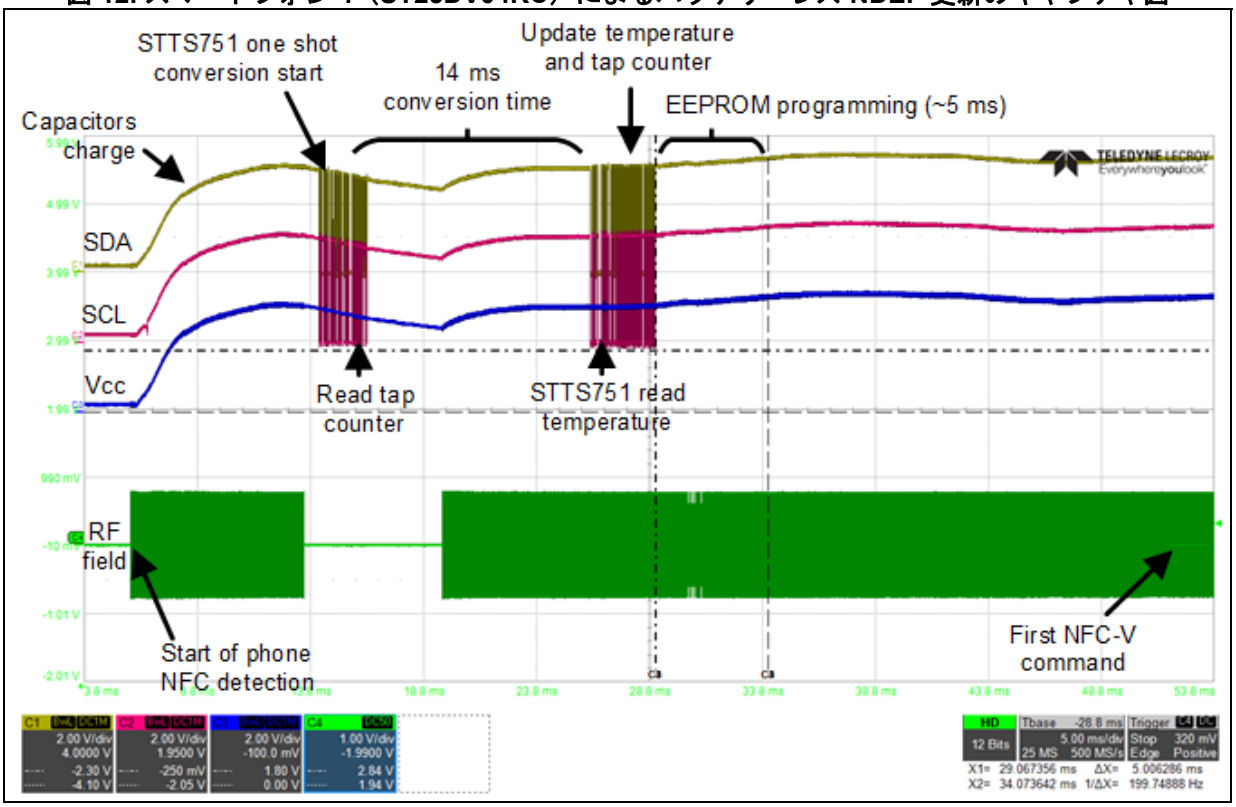
| バイト・アドレス | バイト値 | ASCII | コメント |
|---|---|---|---|
| 0000 | E1 40 40 00 03 2A D1 01 26 55 01 65 78 61 6D 70 | á@@..*Ñ.&U.examp | CC ファイル+TLV |
| 0010 | 6C 65 2E 63 6F 6D 2F 63 75 72 72 65 6E 74 5F 74 | le.com/current_t | - |
| 0020 | 65 6D 70 3D 30 30 30 30 2F 74 63 3D 30 30 30 30 | emp=0000/tc=0000 | 更新されるページ |
| 0030 | FE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | Þ…… | TLV ターミネータ |

ST25DV04KC を使ってマイクロコントローラで実行されるサンプルコード：

```
/* initialize string to be written in NDEF */
char data_char[16] = "emp=0000/tc=0000";
/* initialize I2C bus and clocks */
I2C_Init();
/* Lanch sensor acquisition */
OneShotTemperature ();
/* read current tap counter in tag's EEPROM (4Bytes) */
I2C_ReadBuffer(ST25DV _ADDRESS_USER, 0x002C, 4, data_char+12);
/* wait for sensor to complete acquisition */
Delay(14ms);
/* Read I2C sensor to get current temperature */
ReadOneTemperature (&data_sensor);
/* convert temperature value to ASCII */
ConvertTempToAscii(data_sensor, data_char+4);
/* increment tap counter value in ASCII */
IncrementTapCounterAscii(data_char+12);
/* update temperature and tap counter values in NDEF message (16 bytes) */
I2C_WriteOnePage(ST25DV_ADDRESS_USER, 0x0020, data_char);
/* immediately stop MCU to reduce power consumption */
halt();
```

図 12 は、スマートフォン 4 での NDEF 更新動作を示すオシロスコープのスクリーンショットです。

図 12. スマートフォン 4（ST25DV04KC）によるバッテリ・レス NDEF 更新のキャプチャ図




ST25DV04KC では、EEPROM 内の NDEF 更新終了から最初の NFC-V コマンドまでの間に追加のタイミングマージンが得られます。

### 2.1.5 バッテリ駆動による NDEF 内容自動更新の実装

30ms というタイミング制約が達成できない場合、NDEF メッセージの更新にもっと長い時間をかけられる別の方法があります。

セクション 2.1.2 に見られたように、スマートフォンは短い RF フィールドのバーストを送信することで、タグの存在を継続的にポーリングしています。この方法では、ST25DV-I2C の能力を利用して、NDEF 更新のトリガとなるこれらの RF フィールドのバーストを検出します。NDEF が更新されると、NDEF メッセージが読み取られるまで、ST25DV-I2C は次のスマートフォンのポーリング・シーケンスに正常に応答するように設定されています。このため、スマートフォンがタグをタップしたときに若干の応答遅延が生じますが、ほとんどのスマートフォンではこの遅延は 500ms 以下に制限されています。

2 つのポーリング・シーケンスの間には RF フィールドがないため、この方法ではエナジー・ハーベスティングを使用できず、ST25DV-I2C とマイクロコントローラには外部から電力を供給する必要があります。

原則として、この拡張 NDEF アプリケーションは次のように動作します。

- 標準の NFC タグとして動作する
- RF イベント時にマイクロコントローラをウェイクアップする
- バッテリまたは常設電源を使用して、マイクロコントローラと、可能性のある他のオンボード外部デバイスに電力を供給する
- 最初の RF フィールドの立ち上がり時に NDEF メッセージを更新するRF フィールドがオフになってもバッテリ電力によって NDEF 更新を可能にする

– 次の RF フィールドの立ち上がりで、スマートフォンからの NDEF の読取りを待つ
– スマートフォンによる NDEF の読取りを検出した後、次の RF フィールドの立ち上がりを待って NDEF を再び更新する
– 最初の NDEF メッセージは、ST25DV-I2C メモリに最初にプログラムされている

この拡張 NDEF 法は RF イベント検出に基づいています。アプリケーションでは RF フィールドの立ち上がりイベントと NDEF メッセージの読取りを検出する必要があります。

RF フィールドの立ち上がり検出は、ST25DV-I2C の RF_FIELD_RISING イベント検出を使用して実行されます。

NDEF メッセージの読取りは、RF_ACTIVITY と RF_FIELD_FALLING の 2 つのイベント検出を組み合わせることで解釈できます。RF アクティビティが検出され、RF フィールドがオフになると（スマートフォンが取り外されたことを意味する）、スマートフォンでの NDEF メッセージの読取りが終了します。

NDEF の更新がすでに行われているかどうかを継続的に追跡するためのフラグとしてグローバル変数が使用されます。

– RF フィールドの立ち上がりを検出すると、マイクロコントローラはこの変数をチェックして NDEF の更新が必要かどうかを判断します。フラグがセットされていない場合、NDEF の更新が行われ、フラグがセットされます。
– RF アクティビティと RF フィールドの立ち下がりを検出すると、マイクロコントローラはこのフラグをチェックして NDEF の更新が前に実施されているかどうかを判断します。実施されている場合、NDEF メッセージは読み取られたと見なされ、フラグは解除されます。

NDEF の更新中、ST25DV-I2C は RF_SLEEP モードに設定され、受信 RF コマンドは無視されます。これによって次の 2 つの利点が得られます。

– NDEF 更新時にスマートフォンによるタグの誤検出や部分的な検出を防ぐ
– I2C アクセス中の RF の外乱を防ぐ（I2C アクセスが RF アクセスによってブロックされない）

この方式では、バッテリ・レス方式の場合のティアリングのリスクがないため、ティアリング防止技術を実装する必要はありません（ST25DV-I2C は、EEPROM プログラミング中にバッテリから継続的に電力を供給されます）。

図 13 と図 14 に NDEF の完全な更新シーケンスを要約します。

**図 13. バッテリ駆動による NDEF 更新の時間経過図（1/2）**

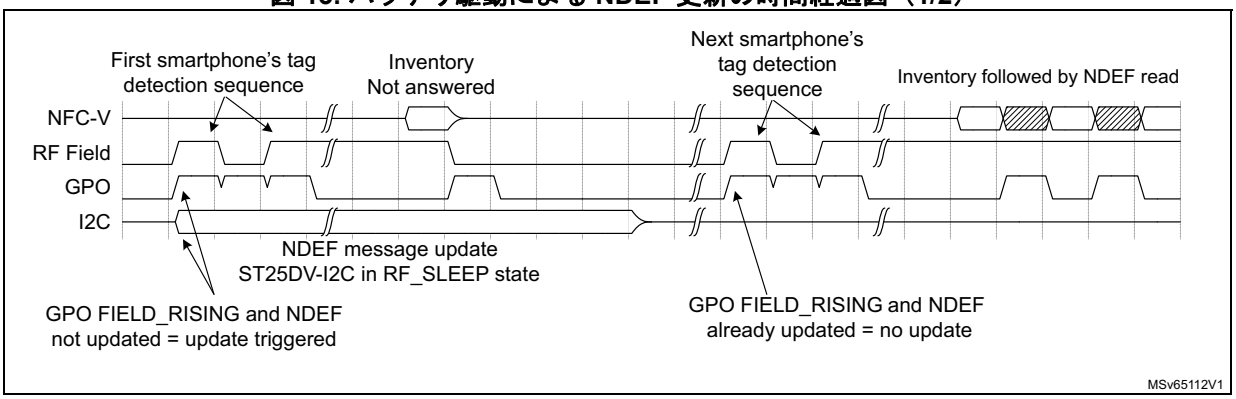
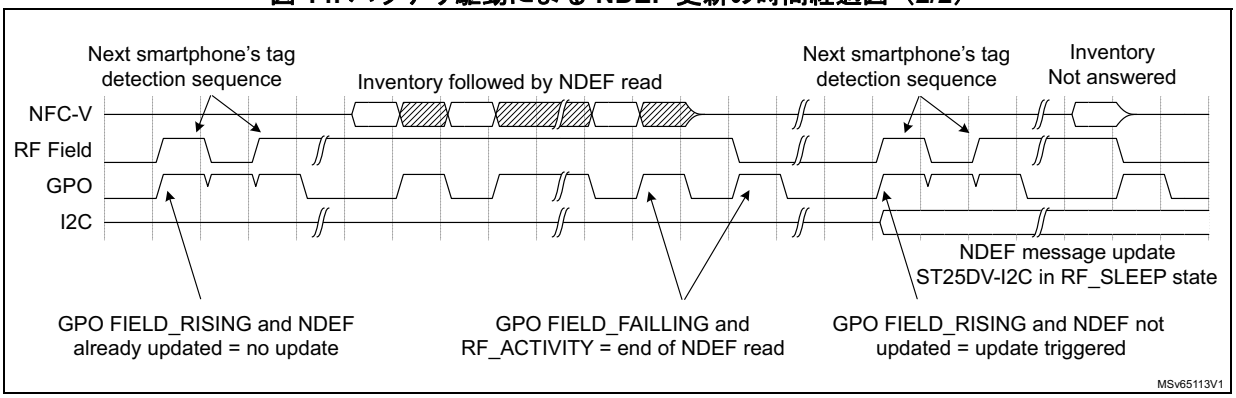


**図 14. バッテリ駆動による NDEF 更新の時間経過図（2/2）**




## バッテリ駆動による NDEF 内容自動更新の実装例

この例では、拡張 NDEF タグから SMS を含む NDEF メッセージが提供されます。このメッセージは、タグの UID、3 つのセンサの値、および NDEF を読み取るたびにインクリメントされるタップ・カウンタで自動的に更新されます。

ハードウェアのセットアップ：

- NDEF メッセージがプリロードされた ST25DV04K ダイナミックタグを組み込んだ X-NUCLEO-NFC04A1 シールド
- STM32L053 マイクロコントローラを内蔵した NUCLEO-L053R8 ボード

ST25DV-I2C メモリは NDEF メッセージでプリロードされ、センサとタップカウンタの値は 0 に設定されています。

マイクロコントローラで実行されるサンプルコード：

```
/* Init ST25DV driver */
while( NFC04A1_NFCTAG_Init(NFC04A1_NFCTAG_INSTANCE) != NFCTAG_OK );
/* Set EXTI settings for GPO Interrupt */
NFC04A1_GPO_Init();
/* Set GPO Configuration: RF Field change and RF activity */
NFC04A1_NFCTAG_ConfigIT(NFC04A1_NFCTAG_INSTANCE,ST25DV_GPO_ENABLE_MASK |
ST25DV_GPO_FIELDCHANGE_MASK | ST25DV_GPO_RFACTIVITY_MASK);

/* main loop */
while(1)
{
  if( GPOActivated == 1 )
  {
    /* Read ITSTS_Dyn to determine source of the GPO interrupt */
    NFC04A1_NFCTAG_ReadITSTStatus_Dyn(NFC04A1_NFCTAG_INSTANCE, &ItStatus )
;
    switch( ItStatus )
    {
    case ST25DV_ITSTS_DYN_FIELDRISING_MASK:
    case ST25DV_ITSTS_DYN_FIELDRISING_MASK |
ST25DV_ITSTS_DYN_FIELDFALLING_MASK:
```

```
    case ST25DV_ITSTS_DYN_FIELDRISING_MASK |
ST25DV_ITSTS_DYN_RFACTIVITY_MASK:
      /* RF field rising detected */
      if( NDEF_update_done == 0) /* update to be done ? */
      {
        /* set ST25DV in RF_SLEEP mode */
        passwd.MsbPasswd = 0; passwd.LsbPasswd = 0;
        NFC04A1_NFCTAG_PresentI2CPassword(NFC04A1_NFCTAG_INSTANCE, passwd
);
        NFC04A1_NFCTAG_SetRFSleep( NFC04A1_NFCTAG_INSTANCE );
        /* update NDEF content */
        MX_NFC4_NDEFUpdate();
        NDEF_update_done = 1;
        /* exit ST25DV from RF_SLEEP mode */
        NFC04A1_NFCTAG_ResetRFSleep( NFC04A1_NFCTAG_INSTANCE );
        passwd.MsbPasswd = 123; passwd.LsbPasswd = 456;
        NFC04A1_NFCTAG_PresentI2CPassword(NFC04A1_NFCTAG_INSTANCE, passwd
);
      }
      break;
    case ST25DV_ITSTS_DYN_FIELDFALLING_MASK |
ST25DV_ITSTS_DYN_RFACTIVITY_MASK:
    case ST25DV_ITSTS_DYN_FIELDRISING_MASK |
ST25DV_ITSTS_DYN_FIELDFALLING_MASK | ST25DV_ITSTS_DYN_RFACTIVITY_MASK:
      /* RF activity and RF field falling detected */
      if( NDEF_update_done == 1) /* update already done ? */
      {
        /* NDEF has been read, next RF field rising: update to be done */
        NDEF_update_done = 0;
      }
      break;
    case ST25DV_ITSTS_DYN_RFACTIVITY_MASK:
    case ST25DV_ITSTS_DYN_FIELDFALLING_MASK:
    default:
        break;
    }
    GPOActivated = 0;
  }
}

/* call back function from IT */
void BSP_GPO_Callback(void)
{
  GPOActivated = 1;
}
```

STM32L0 シリーズは、GPO 割込みによって GPOActivated グローバル変数が 1 に設定されるように構成されています。メインループは、NDEF の更新プロセスを開始するために、この変数値の変化に対してポーリングを行っています。最初のステップは、RF イベントの性質をチェックするために ITSTS_Dyn ステータス・レジスタを読むことです。その後、RF イベントとグローバル変数 NDEF_update_done の値に応じて、NDEF メッセージを更新するか否かが決定されます。

図 15 と図 16 は、スマートフォン 4 での NDEF 更新動作を示すロジック・アナライザのスクリーンショットです。

図 15. スマートフォン 4 によるバッテリ駆動 NDEF 更新のキャプチャ図（NDEF 更新フェーズ）

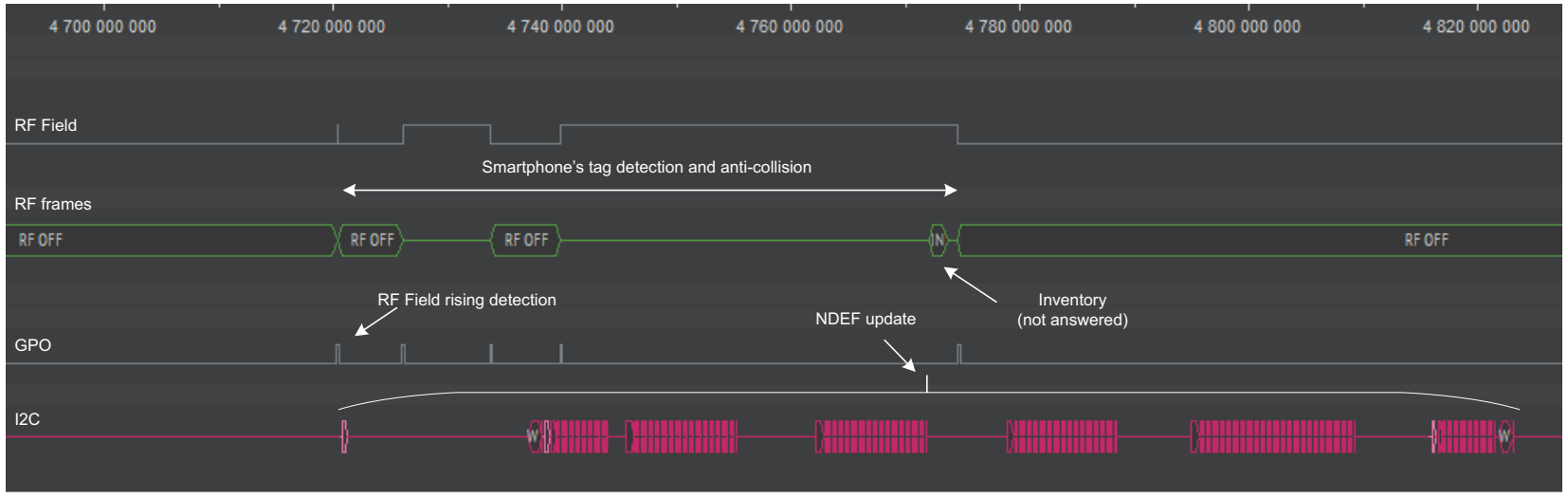
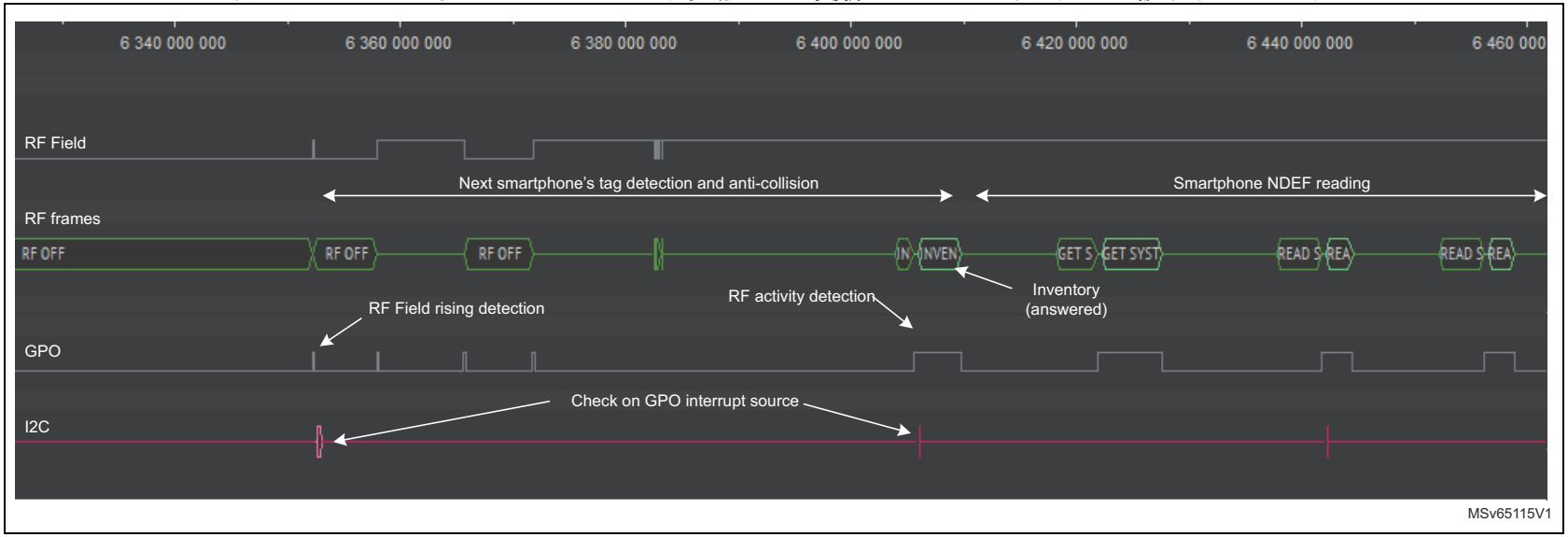
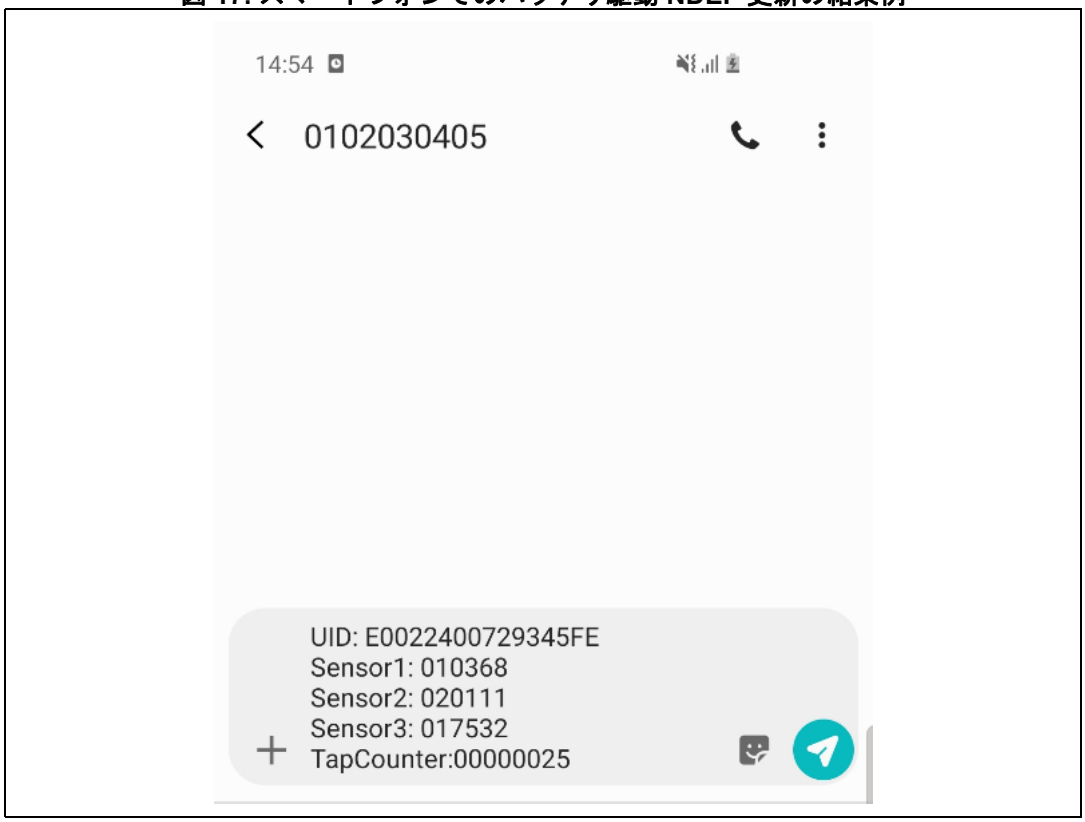



図 16. スマートフォン 4 によるバッテリ駆動 NDEF 更新のキャプチャ図（NDEF 読取りフェーズ）



タグのアンテナにスマートフォンをかざすと、スマートフォンは自動的にメッセージング・アプリケーションを開き、図 17 に示すように、タグの UID、最新のセンサ値、タップ回数が記載された SMS が表示されます。

**図 17. スマートフォンでのバッテリ駆動 NDEF 更新の結果例**




## 2.2 定期的な NDEF の内容更新と外部イベントに基づく更新

NDEF の更新がスマートフォンの存在検出に同期して動的に行えない場合は、非同期に行うことができます。

この場合、NDEF メッセージの内容は、RF アクティビティとは無関係に、定期的に、あるいは他の外部イベントに応じて更新されます。たとえば、アプリケーションでアラームがトリガされるたびに NDEF コンテンツを更新したい場合や、NDEF メッセージがデータログとして使用される場合などに該当します。

この方法では I2C アクセスと RF アクセスが非同期であるため、マイクロコントローラとスマートフォンは同時に ST25DV-I2C にアクセスしようとしますが、これは推奨されません。I2C と RF の衝突を避けるため、NDEF メッセージの内容を更新する間、ST25DV-I2C を RF_SLEEP モードに設定することを推奨します。

この方法は特別な技術を必要としないため、このアプリケーション・ノートでは分析しません。

## 2.3 設定データに関する考慮事項

アプリケーションの設定データは、マイクロコントローラの Flash メモリではなく、NFC の EEPROM に保存することをお勧めします。

EEPROM に設定（センサの有効化、サンプリング周期など）を保存する利点は、アプリケーションに電源が供給されていなくても NFC リーダが直接それらを読み書きできることです。

拡張 NDEF の場合、すべてのダイナミック NDEF パラメータは ST25DV-I2C メモリに格納されます（NDEF メッセージ内で更新する情報やアドレスなど）。その後、リーダは NDEF メッセージ本体を更新することができ、そして同時にバイトのアドレスは常に動的に更新されます。マイクロコントローラは、NDEF メッセージを更新する前に、これらの設定バイトを読み取ります。

これは、ST25DV-I2C のユーザ・メモリに 2 つのエリアを定義することで実現され、エリア 0 には常に CC ファイルと NDEF メッセージが格納され、エリア 1 には設定データが格納されます。両エリアは、必要に応じて異なるパスワードで独立して保護することができます。

# 3 改版履歴

表 4. 文書改版履歴

| 日付 | 版 | 変更内容 |
|---|---|---|
| 2020 年 4 月 10 日 | 1 | 初版発行 |
| 2020 年 4 月 28 日 | 2 | セクション 2.1 : スマートフォン検出時の NDEF 内容の自動更新を更新。 |
| 2021 年 5 月 05 日 | 3 | はじめに、セクション 2.1.3 : NDEF の内容の自動更新のタイミング制約、セクション 2.1.4 : バッテリ・レスの NDEF 内容自動更新の実装 および ST25DV04K によるバッテリ・レスの NDEF 内容自動更新の例を更新。<br>表 2 : タグのメモリ内容（ST25DV04K）を更新。<br>ST25DV04KC によるバッテリ・レスの NDEF 内容自動更新の例を追加。<br>ドキュメント全体で文章を軽微に編集。 |

表 5. 日本語版文書改版履歴

| 日付 | 版 | 変更内容 |
|---|---|---|
| 2023 年 9 月 | 1 | 日本語版　初版発行 |

**重要なお知らせ（よくお読み下さい）**

STMicroelectronics NV およびその子会社（以下、ST）は、ST製品及び本書の内容をいつでも予告なく変更、修正、改善、改定及び改良する権利を留保します。購入される方は、発注前にST製品に関する最新の関連情報を必ず入手してください。ST製品は、注文請書発行時点で有効なSTの販売条件に従って販売されます。

ST製品の選択並びに使用については購入される方が全ての責任を負うものとします。購入される方の製品上の操作や設計に関してSTは一切の責任を負いません。

明示又は黙示を問わず、STは本書においていかなる知的財産権の実施権も許諾致しません。

本書で説明されている情報とは異なる条件でST製品が再販された場合、その製品についてSTが与えたいかなる保証も無効となります。

ST およびST ロゴはSTMicroelectronics の商標です。STの登録商標についてはSTウェブサイトをご覧ください。www.st.com/trademarks
その他の製品またはサービスの名称は、それぞれの所有者に帰属します。

本書の情報は本書の以前のバージョンで提供された全ての情報に優先し、これに代わるものです。

この資料は、STMicroelectronics NV 並びにその子会社(以下ST)が英文で記述した資料（以下、「正規英語版資料」）を、皆様のご理解の一助として頂くためにSTマイクロエレクトロニクス㈱が英文から和文へ翻訳して作成したものです。この資料は現行の正規英語版資料の近時の更新に対応していない場合があります。この資料は、あくまでも正規英語版資料をご理解頂くための補助的参考資料のみにご利用下さい。この資料で説明される製品のご検討及びご採用にあたりましては、必ず最新の正規英語版資料を事前にご確認下さい。ST及びSTマイクロエレクトロニクス㈱は、現行の正規英語版資料の更新により製品に関する最新の情報を提供しているにも関わらず、当該英語版資料に対応した更新がなされていないこの資料の情報に基づいて発生した問題や障害などにつきましては如何なる責任も負いません。